

**AMENDMENTS TO THE SPECIFICATION**


Please amend paragraph [0003] to read:

Voice extensible markup language, or VoiceXML, is a markup language for voice or speech-driven applications. VoiceXML is used for developing speech-based telephony applications, and also enables web-based content to be accessed via voice using a telephone. VoiceXML is being developed by the VoiceXML Forum, ~~and is described at~~ <http://www.voicexml.org>. Due to the verbose nature of VoiceXML, it can be cumbersome to develop VoiceXML-based applications manually using a text or XML editor. Consequently, voice application development systems are available that allow voice applications to be developed by manipulating graphical elements via a graphical user interface rather than coding VoiceXML directly. However, these systems are limited in their ability to assist a developer. It is desired to provide a process and system for developing a voice application that improves upon the prior art, or at least provide a useful alternative to existing voice application development systems and processes.

Please amend paragraph [0054] to read:


To facilitate the reverse translation from VoiceXML code to dialog, the dialog transformer 204 modifies the VoiceXML code by inserting additional attributes into various element tags, providing dialog element information that cannot be stored using the available VoiceXML tags. The resulting file 214 is effectively in an extended VoiceXML format. The additional attributes are stored in a separate, qualified XML namespace so that they do not interfere with the standard VoiceXML elements and attributes, as described in the World Wide Web Consortium's (W3C) *Namespaces in XML* recommendation, ~~available at~~ <http://www.w3.org/TR/1999/REC-xml-names-19990114/>. This facilitates the parsing of extended VoiceXML files.

Please amend paragraph [0091] to read:


<b>Remote Processing</b> 	
<b>Description:</b>	Call a server-side script (via a HTTP URL) to perform some processing. The script should return a VoiceXML document with the results of the processing.

<b>NOTES:</b>	<p>Need PHP/CGI running on remote Web server to handle this. Name of the input and output parameters needn't be declared in the Variables element. The names of the input and output parameters should match what is required at the remote server end. Other Dialog Elements can refer to the output of the Remote Processing element using:          "RemoteProcName.outputName".</p> <p><u>See Chapter 9: Advanced Techniques for more information</u></p>
<b>Parameters:</b>	<p><b>Remote Processing</b></p> <p><b>Name:</b> a name for this Remote Processing element. An object with this name will be created to store the returned outputs. Each output can then be accessed as \$remoteProcessingName.\$output, <i>eg. ValidatePin.validity</i></p> <p><b>Source URL:</b> the URL of the remote script to execute, <i>eg. http://server/cgi-bin/script1.</i></p> <p><b>Input</b></p> <p><b>Name:</b> name of the input parameter to be submitted, <i>eg. price</i>. This is the name that the server script expects.</p> <p><b>Value:</b> value of the input parameter to be submitted, <i>eg. BuyForm.price, 1, 'AUD', true, x + 1</i>. This may be any valid ECMAScript expression. The expression may reference any variable defined in the package, such as in a Variables element or a Form object. If the value is an ECMAScript object with fields f1, f2, ..., then the object is serialised and submitted using the names o.f1, o.f2, ..., assuming o is the declared input name.</p> <p><b>Output</b></p> <p><b>Name:</b> name of each parameter returned by the remote script, <i>eg. outField1</i>. The remote script should return a VoiceXML document containing a subdialog form whose return namelist includes all these output fields.</p>

Please amend paragraph [0092] to read:

<b>Loop Call</b> 	
<b>Description:</b>	Calls a Loop to execute a part of a dialog that should be iterated several times.
<b>Notes:</b>	<p>A corresponding Loop element is required.</p> <p><u>See Chapter 9: Advanced Techniques for more information</u></p>
<b>Parameters:</b>	<p><b>Loop Call</b></p> <p><b>Name:</b> a name for this Loop Call element.</p> <p><b>Source:</b> the loop to call. The loop must be defined in this package.</p>

Please amend paragraph [0093] to read:

<b>Subroutine Call</b> 	
<b>Description:</b>	Start a subroutine. Used for breaking bigger programs into smaller components, for ease of readability, reuse of code, and support of pre-packaged code.
<b>Notes:</b>	A corresponding Subroutine element is required. Input parameter names needn't be declared.

	<p>Output parameter names are set by the Subroutine element. Other Dialog Elements can refer to the output parameters by calling: "SubroutineCallName.OutputParamName".</p> <p><del>See Chapter 9: Advanced Techniques for more information</del></p>
<b>Parameters:</b>	<p><b>Subroutine Call</b></p> <p><b>Name:</b> a name for this Subroutine Call element. An object with this name will be created to store the returned outputs. Each output can then be accessed as \$subName.\$output, eg. <i>ValidatePin.validity</i>.</p> <p><b>Source:</b> the subroutine to call (qualified with the package name). If the declared subroutine inputs and outputs have changed, you may need to reselect the subroutine from the Source list box to refresh the displayed inputs and outputs.</p> <p><b>Inputs</b></p> <p>This is a list of the inputs that the called subroutine expects. For each input, you must specify an ECMAScript expression whose value will be passed to the subroutine, eg. <i>0</i>, <i>BuyForm.price</i>, <i>pinValidity</i>, <i>'AUD'</i>. It may reference any variable defined in the package, such as in a Variables element or a Form object. Note that strings should be quoted in single quotes. Note also that you cannot leave any input value blank.</p> <p><b>Outputs</b></p> <p>This is a list of the outputs that the called subroutine returns.</p>

Please amend paragraph [0102] to read:

<b>Subroutine</b>	
<b>Description:</b>	This is the entry point for a subroutine.
<b>Notes:</b>	<p>Subroutine elements cannot be added to the Main Package. They can only be added to non-Main Packages. More than one Subroutine can be added to a non-Main Package. However, all Subroutine inputs and outputs must be unique within the Package, ie. no two Subroutines can declare the same input name or output name. Furthermore, no input name can be the same as an output name.</p> <p>The Subroutine element is created by (1) right-clicking on the "Project" icon on the Dialogs/Project Files window (LHS) to add a new Package; and (2) right-clicking on a Package icon on the Dialogs window (LHS) to add a new Subroutine.</p> <p>Each path in the Subroutine should end with a Return element, otherwise the Subroutine will not return to the calling dialog.</p> <p><del>See Chapter 9: Advanced Techniques for more information</del></p>
<b>Parameters:</b>	<p><b>Subroutine</b></p> <p><b>Name:</b> a name for this Subroutine element.</p> <p><b>Input</b></p> <p><b>Name:</b> name of each input parameter expected, eg. <i>pin</i>. The input will be declared as a document variable that you can access from anywhere within the package. You do not need to (re)declare it under the Variables element.</p> <p><b>Output</b></p> <p><b>Name:</b> name of each return parameter, eg. <i>validity</i>. The output will be declared as a document variable that you can access from anywhere within the package. You do not need to (re)declare it under the Variables element.</p>

Please amend paragraph [0103] to read:

<b>Loop</b>	
<b>Description:</b>	A portion of the dialog that should be executed several times.
<b>Notes:</b>	<p>The Loop element is created by right-clicking on the Package icon on the Dialogs window (LHS). Variables are freely shared between the Loop body and the main dialog (of the same Package) as they are in one VoiceXML document.</p> <p>See Chapter 9: Advanced Techniques for more information</p>
<b>Parameters:</b>	<p><b>Loop</b></p> <p><b>Name:</b> a name for this Loop element.</p> <p><b>Loop Test:</b> an ECMAScript expression that if evaluated to true will cause execution of the next iteration of the loop, eg. counter &lt; 5.</p> <p><b>Test at start:</b> If enabled, the test condition is evaluated at the start of the loop and the loop is equivalent to a while / for loop. If disabled, the loop body is executed before the test condition is evaluated and the loop is equivalent to a do-while loop.</p> <p><b>Exit message:</b> a message to be played when the loop exits normally, eg. there are no more items. The system can generate a default text prompt and a default audio file name for recording the corresponding audio prompt, or you may enter in your own values. If a message is not required, the "Generate default..." checkboxes should be unchecked and the "Audio file" and "TTS" text boxes should be left empty.</p> <p><b>Loop Init</b></p> <p>Variables to be initialised at the start of the loop.</p> <p><b>Name:</b> name of a variable, eg. counter. The variable must have been created elsewhere, such as in the Variables element.</p> <p><b>Value:</b> an ECMAScript expression that sets the initial value for the variable, eg. 0.</p> <p><b>Loop Step</b></p> <p>Variables to be incremented before another iteration of the loop. The increment occurs before the test condition is reevaluated.</p> <p><b>Name:</b> name of a variable, eg. counter.</p> <p><b>Value:</b> an ECMAScript expression to increment the variable, eg. counter + 1.</p>